Converging HTC-HPC: Two Case Studies

Shantenu Jha, Matteo Turilli
For the BigPanDA Team (BNL, UTA, ORNL)

Rutgers Advanced DIstributed Cyberinfrastructure & Applications Laboratory (RADICAL)

http://radical.rutgers.edu

Outline

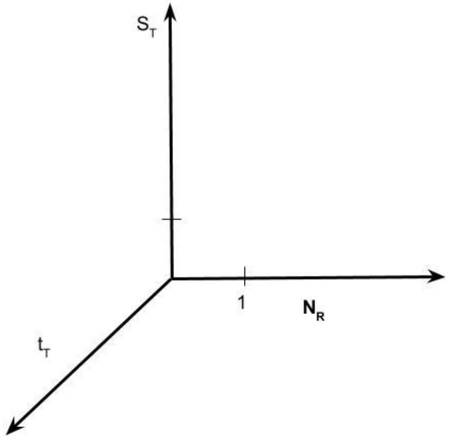
- Understanding the space of “high-throughput” computing
- Abstractions and Execution Models
- Case Study: ATLAS on Titan
  - Overview of PanDA
  - PanDA on Titan
- Case Study: Ensemble-based bio-molecular simulations

High-throughput Computing

- High-throughput is:
  - When the workload is comprised of multiple tasks.
  - Workload of a single application or a single workflow or multiple workflows as part of a computational campaign
  - Typically characterized by the number of tasks executed
- High-throughput computing can also have:
  - Temporal dimension (e.g., minimise the time to execute tasks)
  - Concurrency dimension (e.g., maximise the number of tasks executed)
- High-throughput computing is not:
  - Confined to single core (single node) tasks
  - Only embarrassingly parallel set of tasks

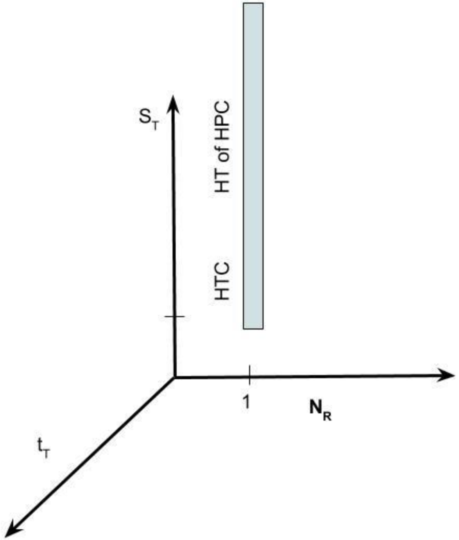
High-throughput Computing

- Parameters that need to be understood:
  - N_T = Number of tasks
  - N_R = Number of resources
  - S_T = Size of each task (#cores)
  - t_T = Duration of each task (modulo performance of resource)
  - N_J = Num. of jobs (land system scheduler)



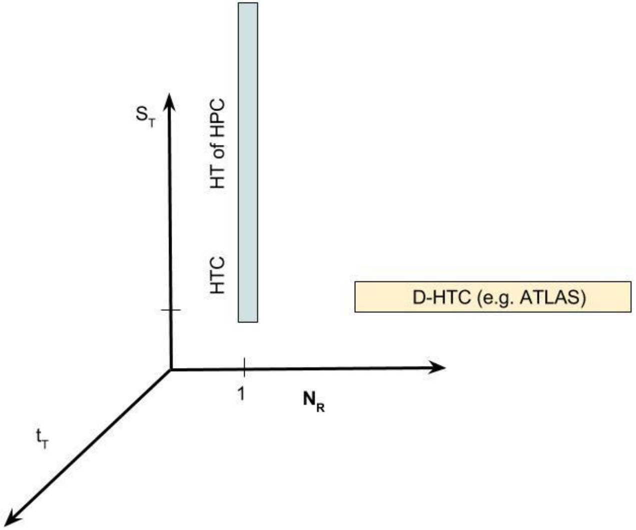
High-throughput Computing

- Parameters that need to be understood:
  - N_T = Number of tasks
  - N_R = Number of resources
  - S_T = Size of each task (#cores)
  - t_T = Duration of each task (modulo performance of resource)
  - N_J = Num. of jobs (land system scheduler)
- (traditional) HTC:
  - ○ $N_T > 1$, $N_T \geq N_R$, $N_R = 1$



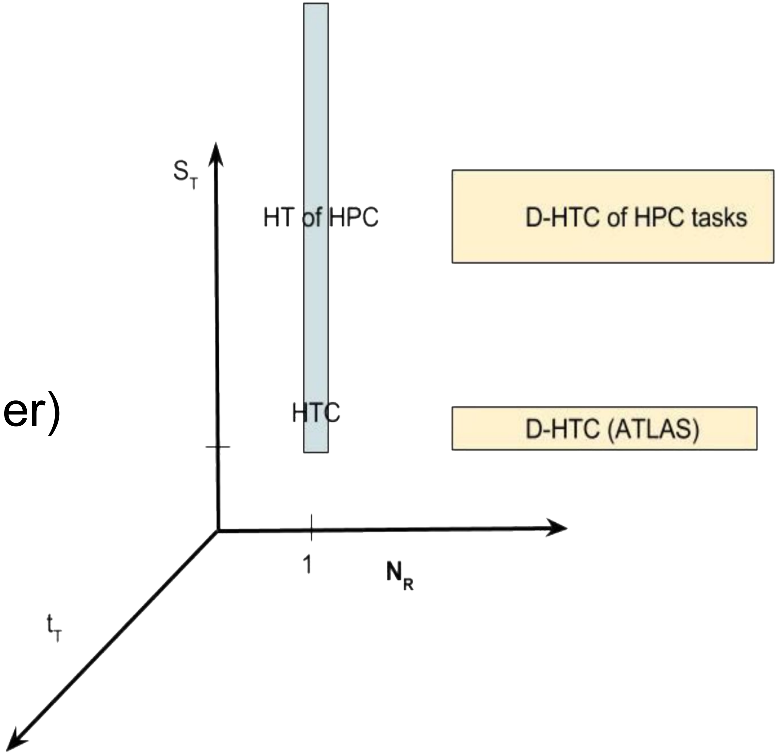
High-throughput Computing

- Parameters that need to be understood:
  - N_T = Number of tasks
  - N_R = Number of resources
  - S_T = Size of each task (#cores)
  - t_T = Duration of each task (modulo performance of resource)
  - N_J = Num. of jobs (land system scheduler)
- (traditional) HTC:
  - ○ $N_T > 1$, $N_T \geq N_R$, $N_R = 1$
- Regime of distributed HTC (D-HTC)
  - ○ $S_T = 1$, $N_R > 1$ ($t_T < 12h$)



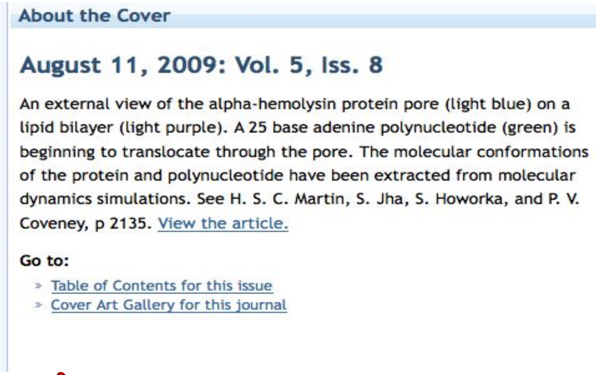
High-throughput Computing

- Parameters that need to be understood:
  - N_T = Number of tasks
  - N_R = Number of resources
  - S_T = Size of each task (#cores)
  - t_T = Duration of each task (modulo performance of resource)
  - N_J = Num. of jobs (land system scheduler)
- (traditional) HTC:
  - $N_T > 1$, $N_T \geq N_R$, $N_R = 1$
- Regime of distributed HTC (D-HTC)
  - $S_T = 1$, $N_R > 1$ ($t_T <$ 12h)
- Regime of D-HTC of HPC tasks
  - $S_T > 1$, $N_R > 1$



Example of D-HTC of HPC tasks

- Developed algorithm to exploit distributed task-level parallelism.
- **2005-09**: Tried running O(1000) simulations on many supercomputers. ***Did not work!*** *Many reasons!*
- Reverted running parallel simulations, sequentially on single resources.
- Importance of **MODELS of EXECUTION,** i.e., reason which tasks go where and when?



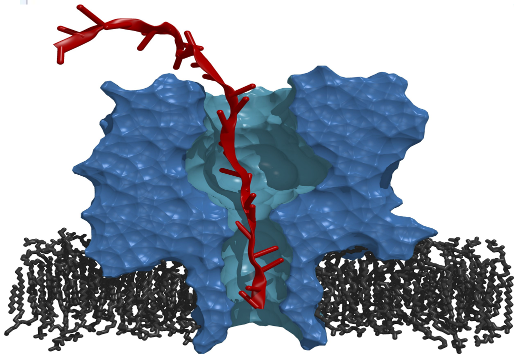
About the Cover

August 11, 2009: Vol. 5, Iss. 8

An external view of the alpha-hemolysin protein pore (light blue) on a lipid bilayer (light purple). A 25 base adenine polynucleotide (green) is beginning to translocate through the pore. The molecular conformations of the protein and polynucleotide have been extracted from molecular dynamics simulations. See H. S. C. Martin, S. Jha, S. Howorka, and P. V. Coveney, p 2135. View the article.

Go to:

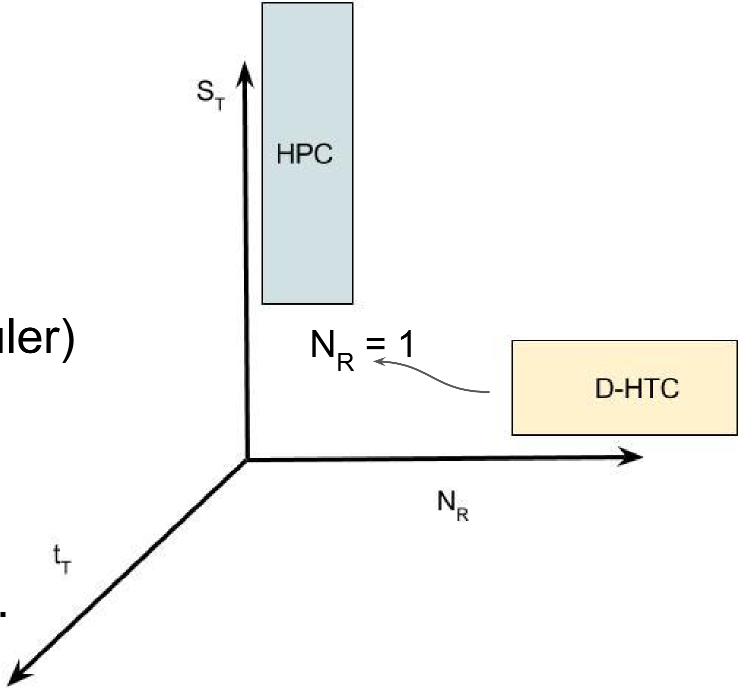
» Table of Contents for this issue
» Cover Art Gallery for this journal



$N_T = 1000, t_T > 12hrs, S_T = 128\text{-}256$ cores

Convergence

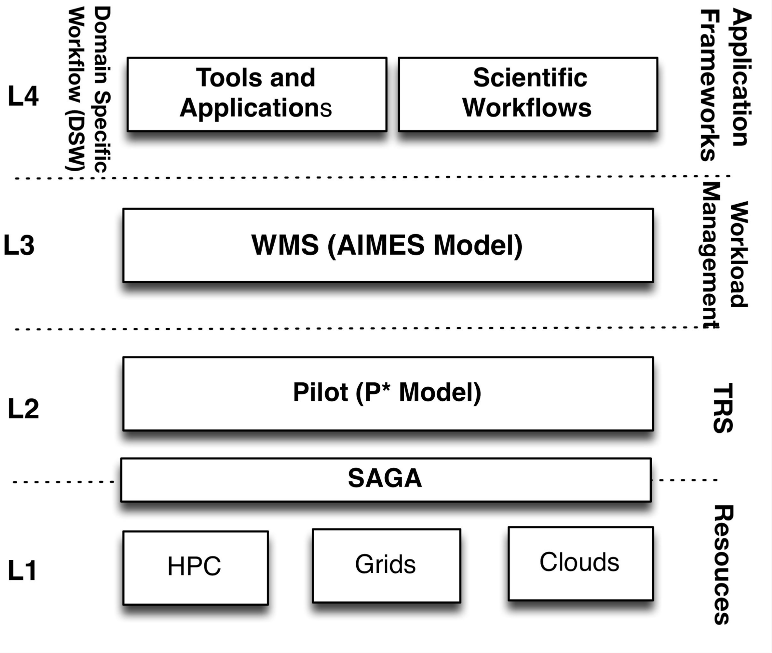
- Parameters that need to be understood:
  - N_T = Number of tasks
  - N_R = Number of resources
  - S_T = Size of each task (#cores)
  - t_T = Duration of each task (modulo performance of resource)
  - N_J = Number of jobs (hit system scheduler)
- **Convergence** due to change of platform:
  - $N_R > 1$, $S_T = 1$
    - → $N_R = 1$ (HPC), $S_T = 1$
  - D-HTC to HTC of HPC tasks (HT-HPC).
  - Change in the number of N_J



Abstractions and Models of Adaptive Execution on Dynamical Resources

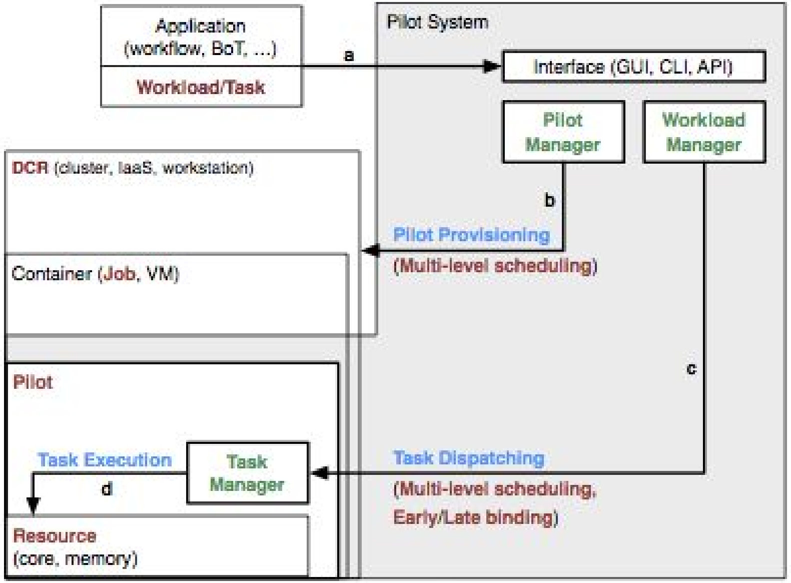
HTC: Requires Support at Multiple Levels

- **Supporting high-throughput computing requires abstraction and models at multiple levels**
- Four Layers:
  - L4: Application
  - L3: Workload Management (**WLMS**)
  - L2: Task Run-time (**TRS**)
  - L1: Resource Access Layer
- For L2: we discuss Pilot abstraction that has proven fundamental for HTC and show its generalization to HPC.



Pilot-Abstraction (P* Model)

- *".. a scheduling overlay which generalizes the reoccurring concept of utilizing a placeholder as a container for compute tasks"*
- Decouples workload from resource management
- Enables fine-grained "slicing and dicing" of resources
- Tighter temporal control, advantages of application-level scheduling
- Build higher-level frameworks without explicit resource management



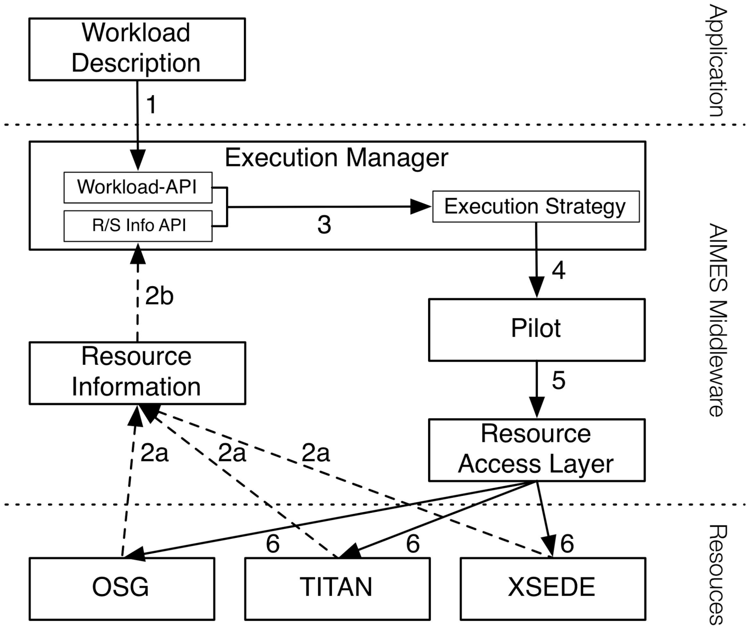
- **"P*: A Model of Pilot-Abstractions"**, ***8th IEEE International Conference on e-Science (2012)***
- ***A Comprehensive Perspective on Pilot-Jobs http://arxiv.org/abs/1508.04180 (2015)***

AIMES Execution Model

- Is there an **execution model** for dynamically federated heterogeneous resources that works **independent** of type of infrastructural dynamism and heterogeneity?

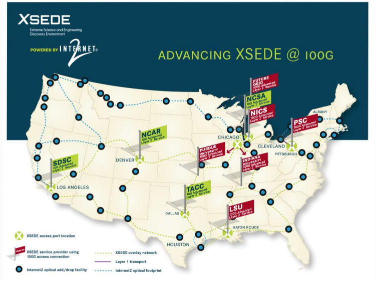
- **AIMES Model** of Execution:
  - Importance of **dynamic integration** of workload and resource information.
  - **Execution strategy**: Temporally ordered set of decisions that need to be made to execute a given workload.

- Conceptual models will → better **implementation**
  - **RADICAL-WLMS** (implementing AIMES model and Pilots) supports **uniform execution models over XSEDE and OSG!**



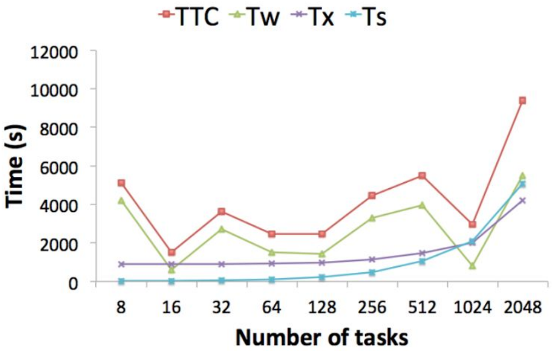
*Schematic of **RADICAL-WLMS** (implementation of AIMES model) approach to workload-resource integration: Evaluate workload requirements & resource capabilities, derive an execution strategy, and enact.*

Distributed CyberInfrastructure: XSEDE

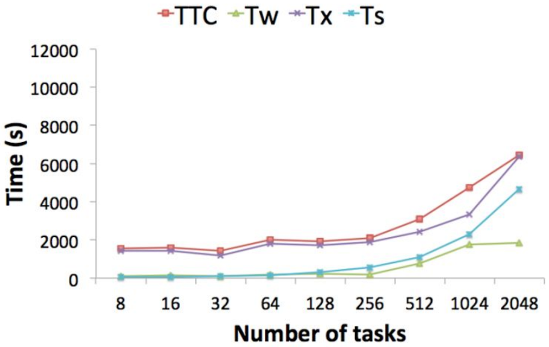

XSEDE
Extreme Science and Engineering Discovery Environment
POWERED BY INTERNET2®
ADVANCING XSEDE @ 100G
FUTURE GRID 100G Advanced Layer 2 Service
NCSA 10G Advanced Layer 2 Service
NICS 100G Advanced Layer 2 Service
PSC 100G Advanced Layer 2 Service
NCAR 10G Advanced Layer 2 Service
SDSC 10G Advanced Layer 2 Service
PURDUE UNIVERSITY 100G Advanced Layer 2 Service
INDIANA UNIVERSITY 100G Advanced Layer 2 Service
TACC 10G Advanced Layer 2 Service
LSU 100G Advanced Layer 2 Service
ALBANY
CHICAGO
CLEVELAND
PITTSBURGH
DENVER
LOS ANGELES
DALLAS
HOUSTON
BATON ROUGE
XSEDE access port location
XSEDE service provider using 100G access connection
Internet2 optical add/drop facility
XSEDE overlay network
Layer 1 transport
Internet2 optical footprint

XSEDE - HPC

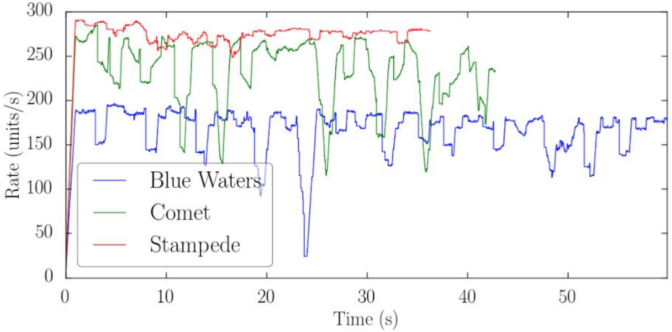
- ~10 supercomputers: ~15 PFLOP
- ~10 storage facilities: ~100 PByte
- shared allocation process & user support
- tailored toward grand challenge applications (HPC)



a

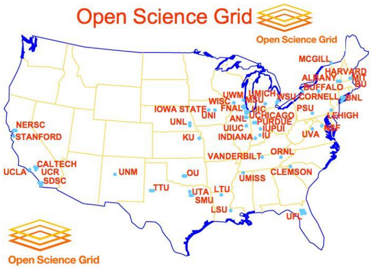


b



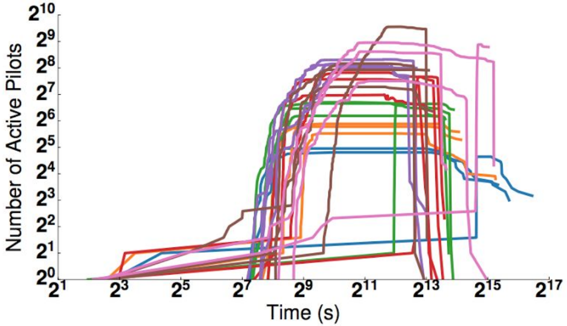
c

Distributed CyberInfrastructure: OSG

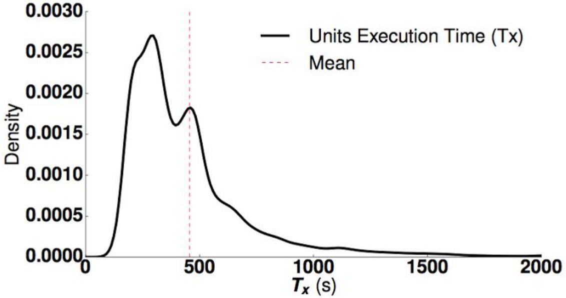


OSG Connect/XSEDE-vcluster - HTC

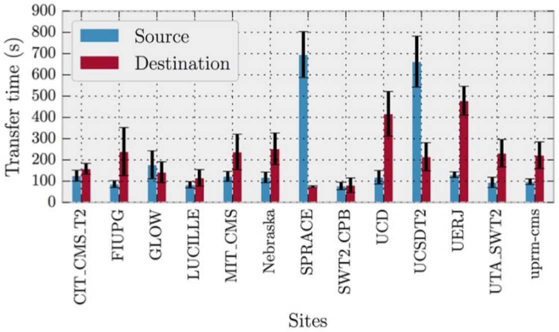
- ~100 sites, ~40,000 cores, ~1 PB   (2012)
- exact composition very dynamic
- only single core machines, loosely coupled
- tailored towards high throughput (HTC)



a



b

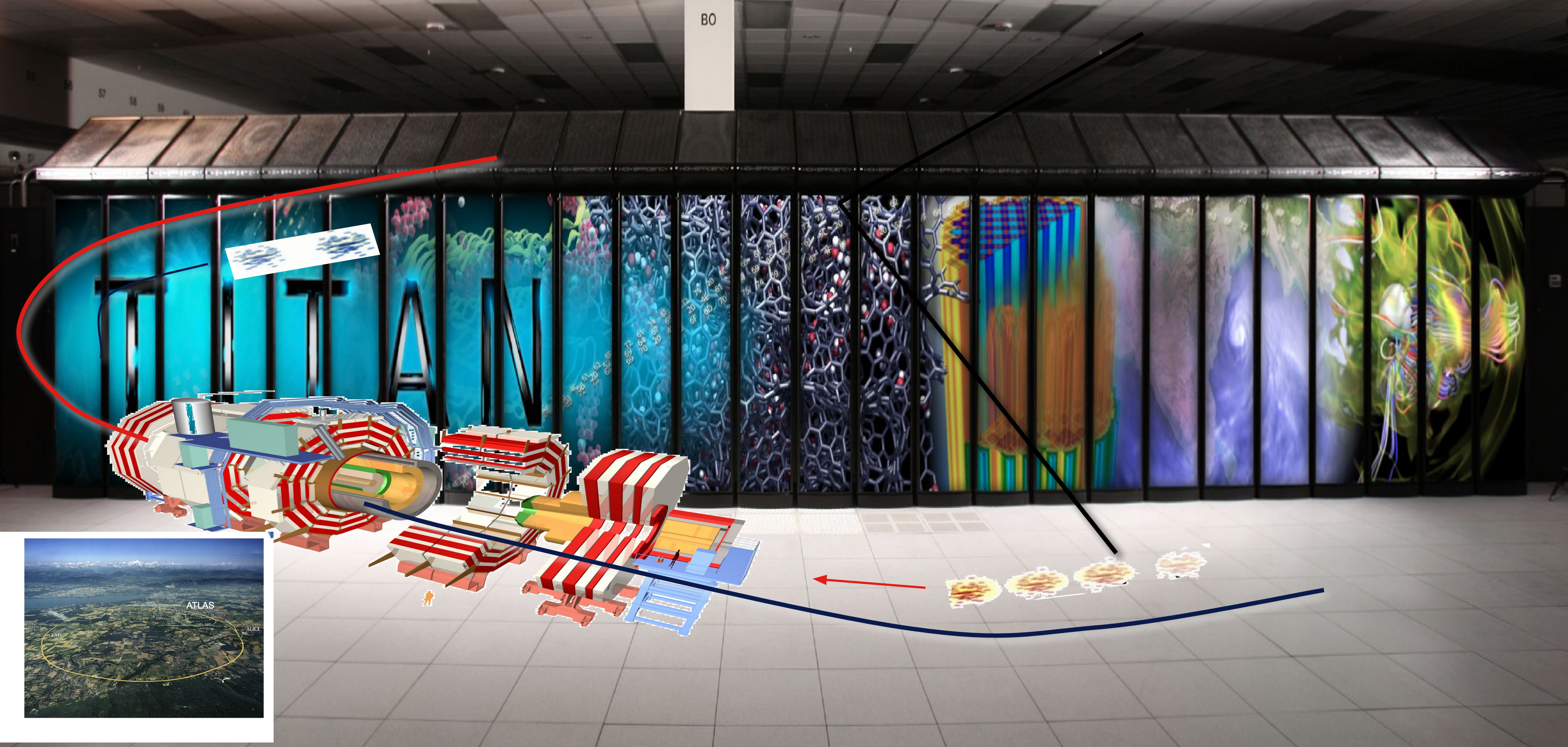


c

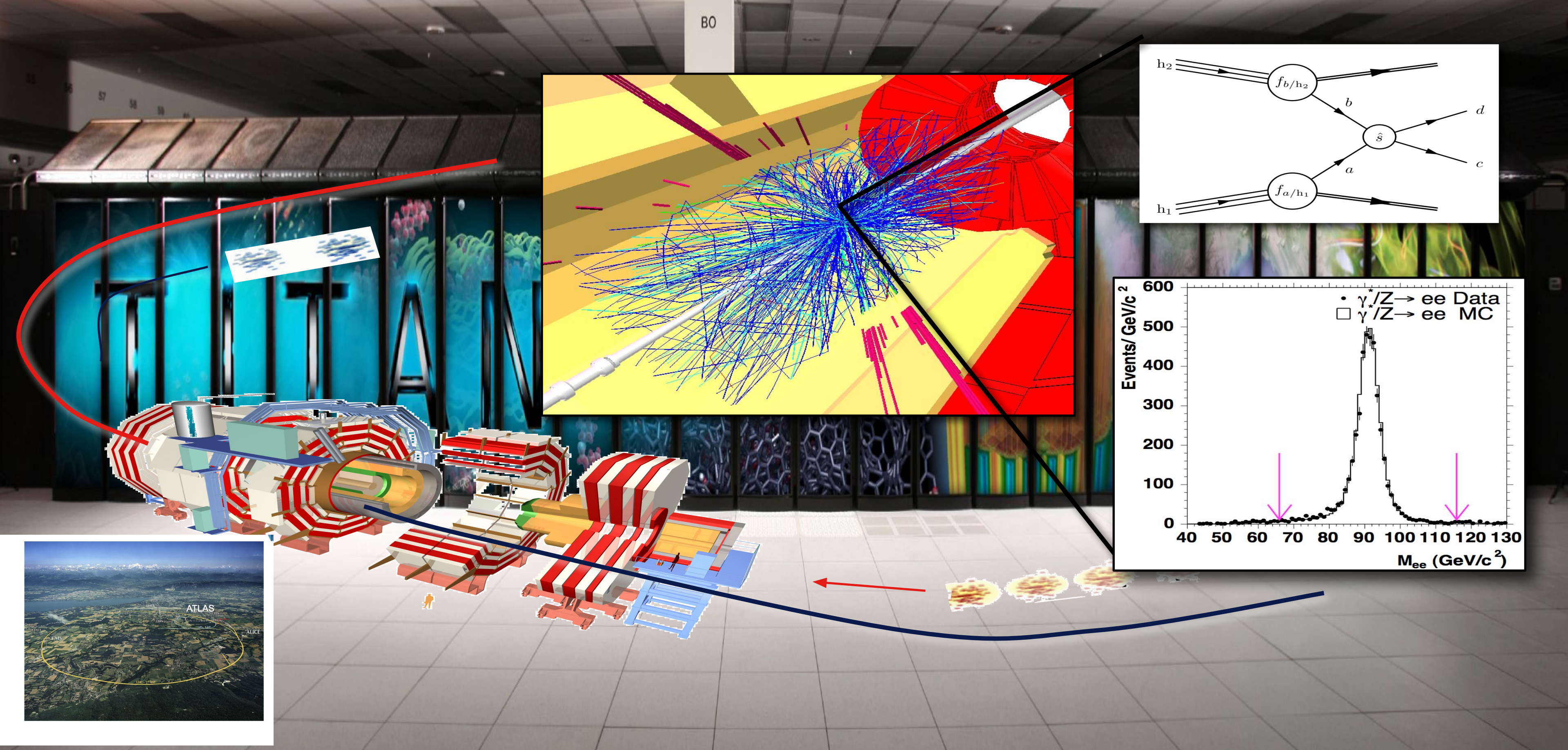
ATLAS: PanDA WMS and HPC Resources Adoption

http://arxiv.org/abs/1704.00978

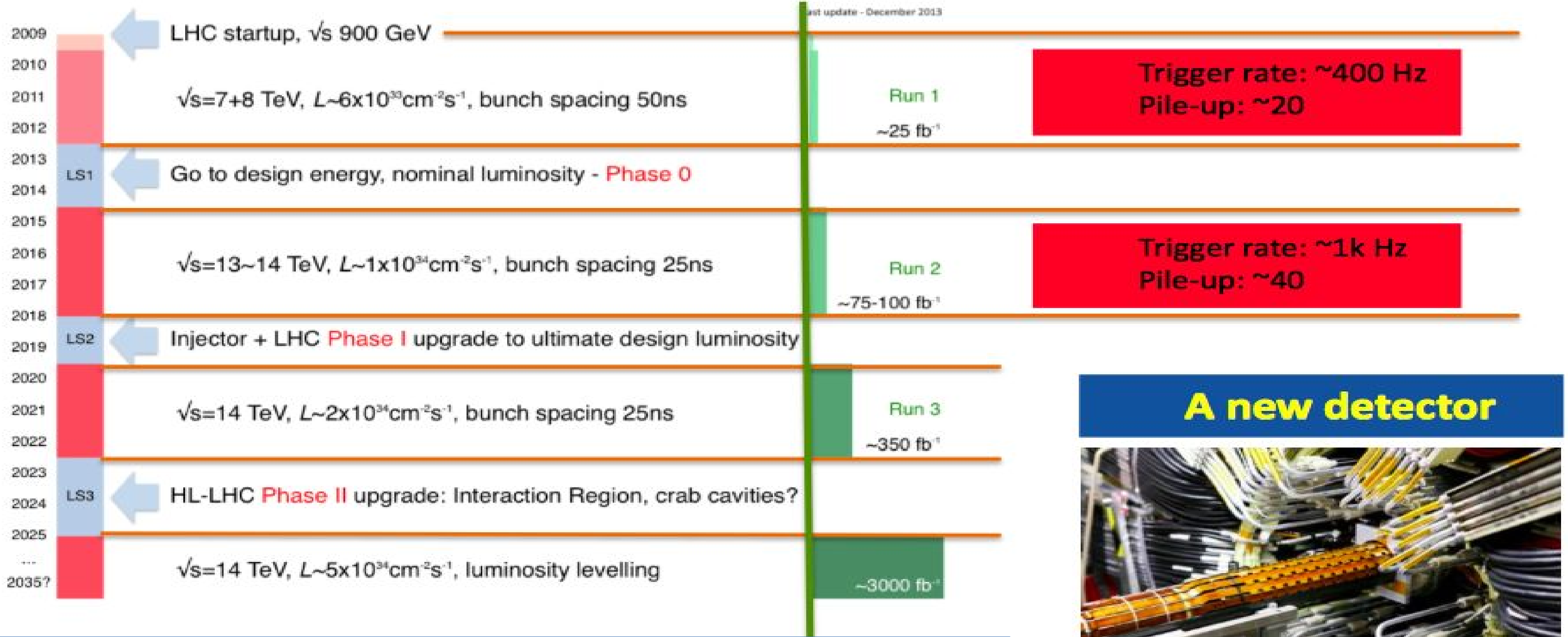
Workflow Management on Titan:
Next Generation Workflow Management for High Energy Physics



Workflow Management on Titan:
Next Generation Workflow Management for High Energy Physics



LHC Upgrade Timeline



Last update - December 2013

2009 — LHC startup, $\sqrt{s}$ 900 GeV

2010–2012 — $\sqrt{s}$=7+8 TeV, $L\sim6\times10^{33}$cm^{-2}s^{-1}, bunch spacing 50ns — Run 1 — ~25 fb^{-1}

Trigger rate: ~400 Hz
Pile-up: ~20

2013–2014 — LS1 — Go to design energy, nominal luminosity - Phase 0

2015–2018 — $\sqrt{s}$=13~14 TeV, $L\sim1\times10^{34}$cm^{-2}s^{-1}, bunch spacing 25ns — Run 2 — ~75-100 fb^{-1}

Trigger rate: ~1k Hz
Pile-up: ~40

2018–2019 — LS2 — Injector + LHC Phase I upgrade to ultimate design luminosity

2020–2022 — $\sqrt{s}$=14 TeV, $L\sim2\times10^{34}$cm^{-2}s^{-1}, bunch spacing 25ns — Run 3 — ~350 fb^{-1}

2023–2025 — LS3 — HL-LHC Phase II upgrade: Interaction Region, crab cavities?

2025 ... 2035? — $\sqrt{s}$=14 TeV, $L\sim5\times10^{34}$cm^{-2}s^{-1}, luminosity levelling — ~3000 fb^{-1}

A new detector

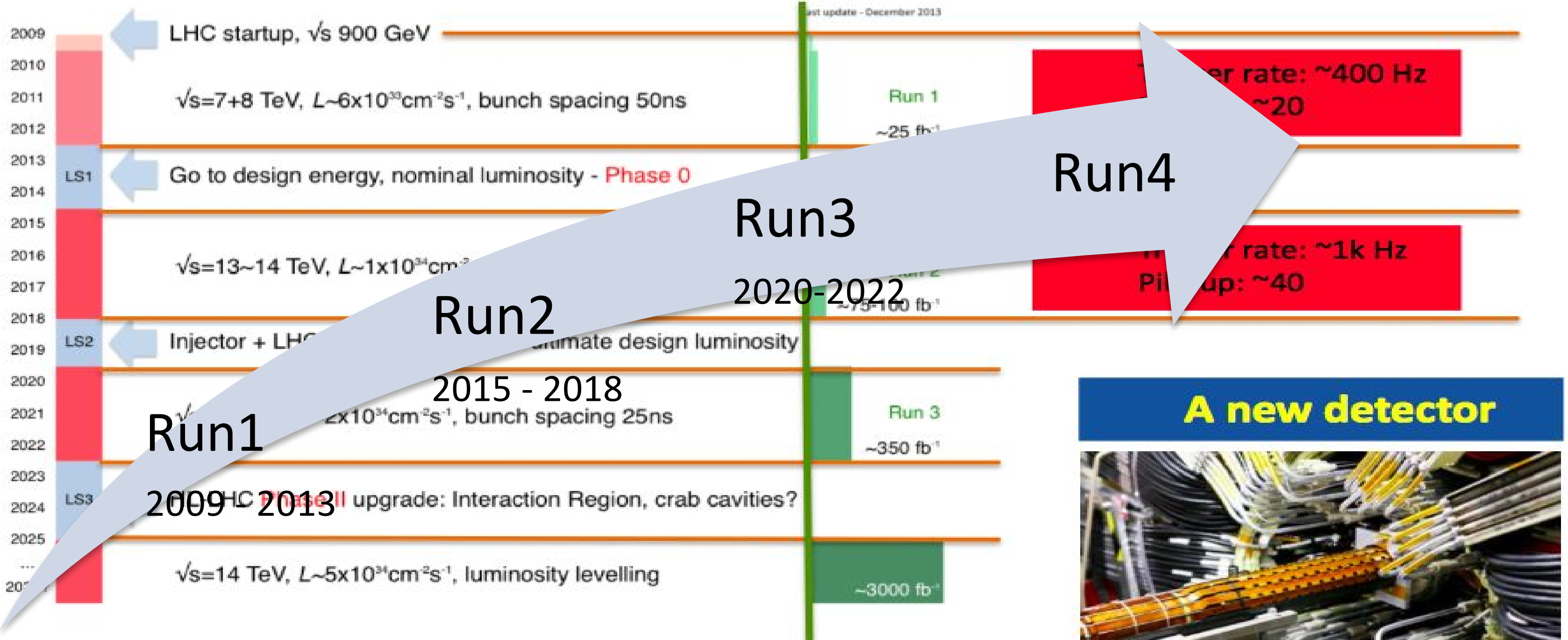
e.g. tracking, calorimeters

In 10 years, increase by factor 10 the LHC luminosity

→ More complex events

→ More Computing Capacity

LHC Upgrade Timeline


Last update - December 2013
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
...
LS1
LS2
LS3
LHC startup, √s 900 GeV
√s=7+8 TeV, L~$6x10^{33}cm^{-2}s^{-1}$, bunch spacing 50ns
Go to design energy, nominal luminosity - Phase 0
√s=13~14 TeV, L~$1x10^{34}cm^{-2}$
Injector + LH
ultimate design luminosity
$2x10^{34}cm^{-2}s^{-1}$, bunch spacing 25ns
HL Phase II upgrade: Interaction Region, crab cavities?
√s=14 TeV, L~$5x10^{34}cm^{-2}s^{-1}$, luminosity levelling
Run 1
~25 fb^{-1}
~75-100 fb^{-1}
Run 3
~350 fb^{-1}
~3000 fb^{-1}
er rate: ~400 Hz
~20
r rate: ~1k Hz
Pi up: ~40
Run1
2009 - 2013
Run2
2015 - 2018
Run3
2020-2022
Run4

A new detector

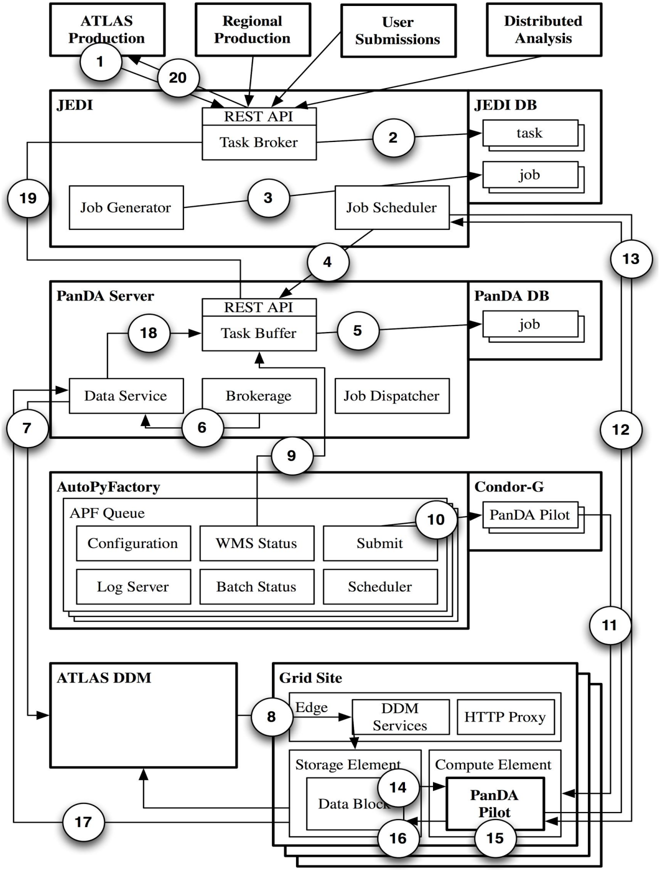
e.g. tracking, calorimeters

In 10 years, increase by factor 10 the LHC luminosity

→ More complex events

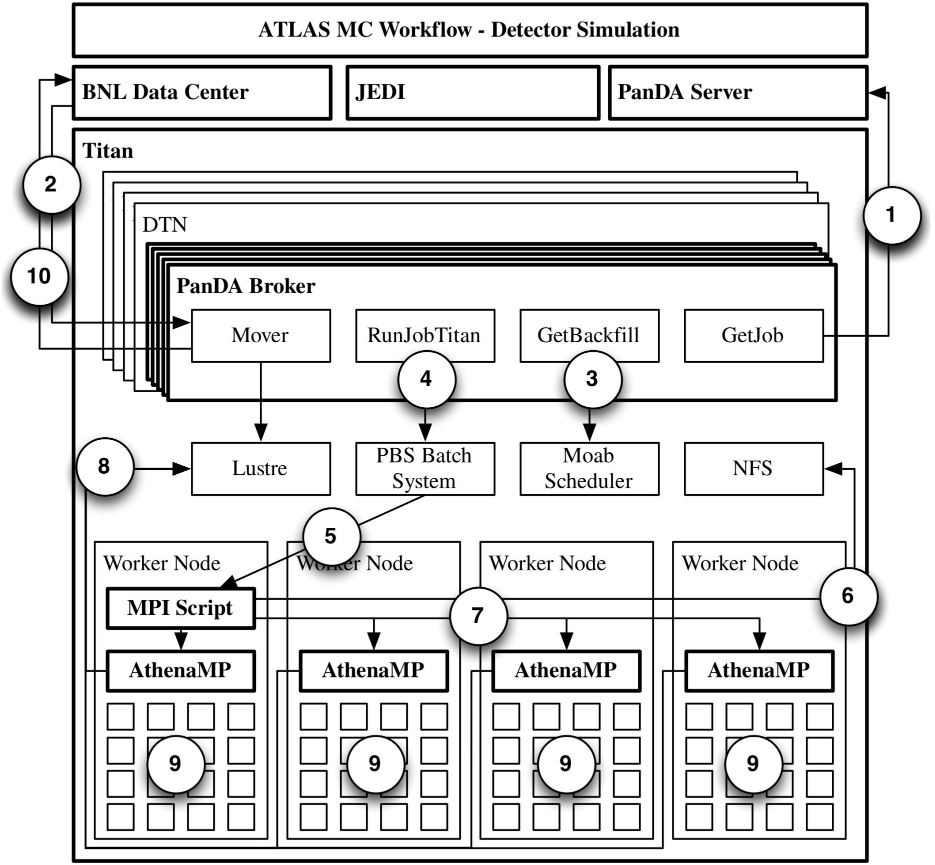
→ More Computing Capacity

Production and Distributed Analysis (PanDA)



- 6 main subsystems:
  - PanDA Server (Run 1)
  - PanDA Pilot (Run 1)
  - JEDI (Run 2)
  - AutoPyFactory (Run 1)
  - PanDA Monitoring (Run 1)
  - Event Service (Run 2)
- Developed for
  - HEP workloads and workflows
  - HTC on Grid infrastructures
- Task-based and job-based execution mode
- Support single researchers, research groups, ATLAS production workflows.

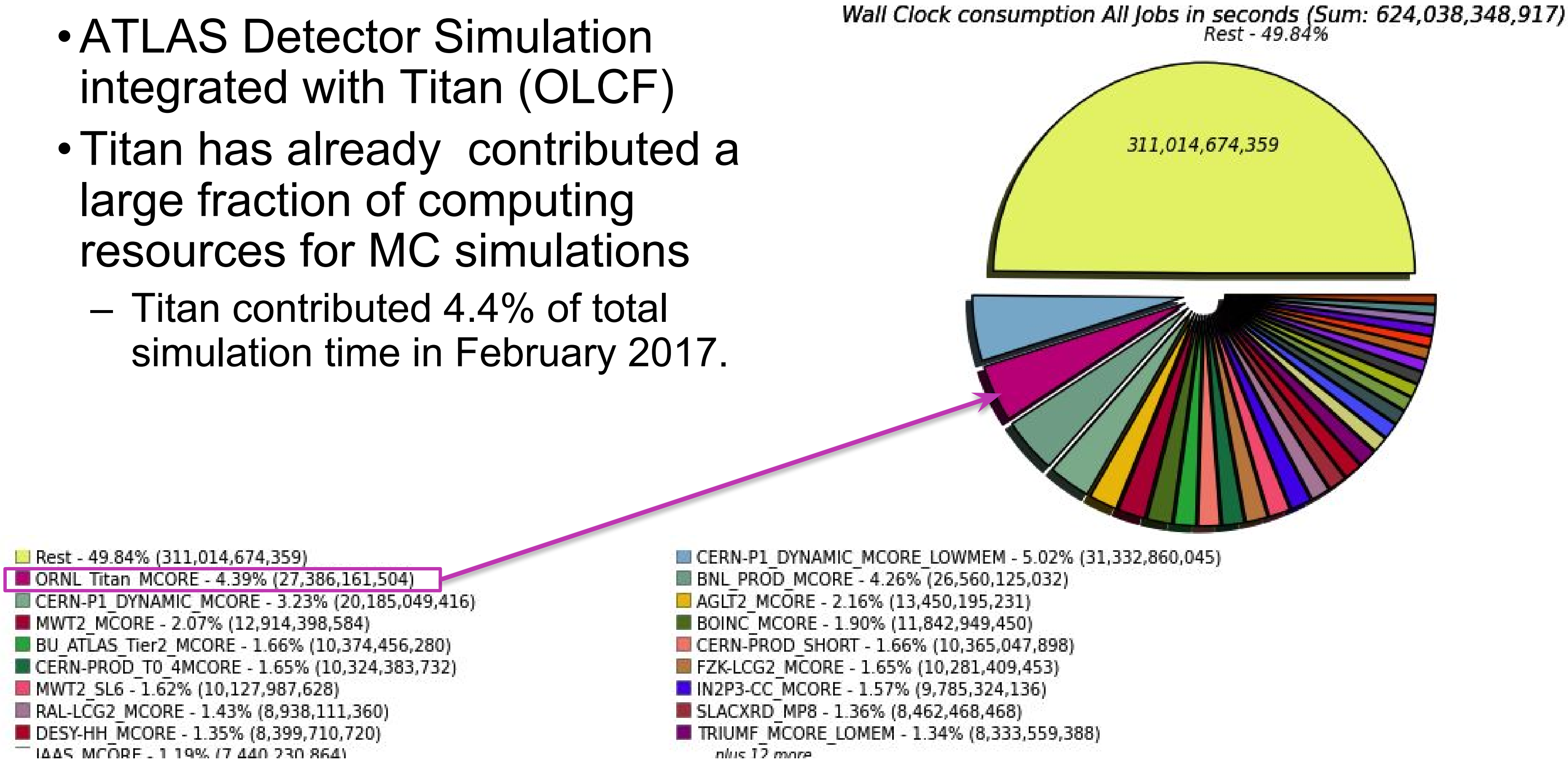
PanDA on Titan



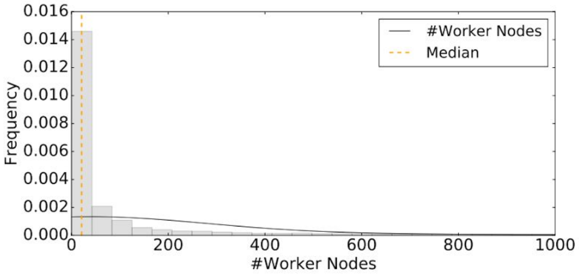
- Cannot deploy PanDA Pilot on Titan’s worker nodes (WN)
- PanDA Brokers on Titan’s DTN
- MPI scripts wrap jobs’ payload
- AthenaMP on read-only NFS shared between DTN and WN
- Events on OLCF Lustre FS

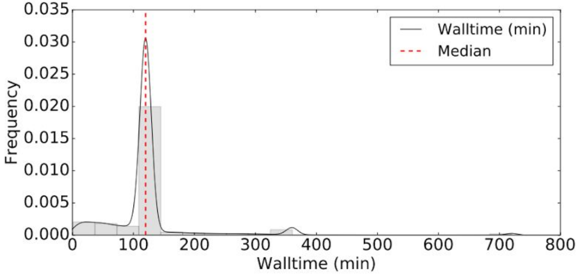
ATLAS simulation time worldwide: February 2017

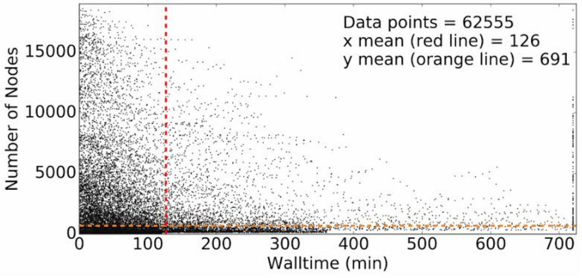
- ATLAS Detector Simulation integrated with Titan (OLCF)
- Titan has already contributed a large fraction of computing resources for MC simulations
  - Titan contributed 4.4% of total simulation time in February 2017.



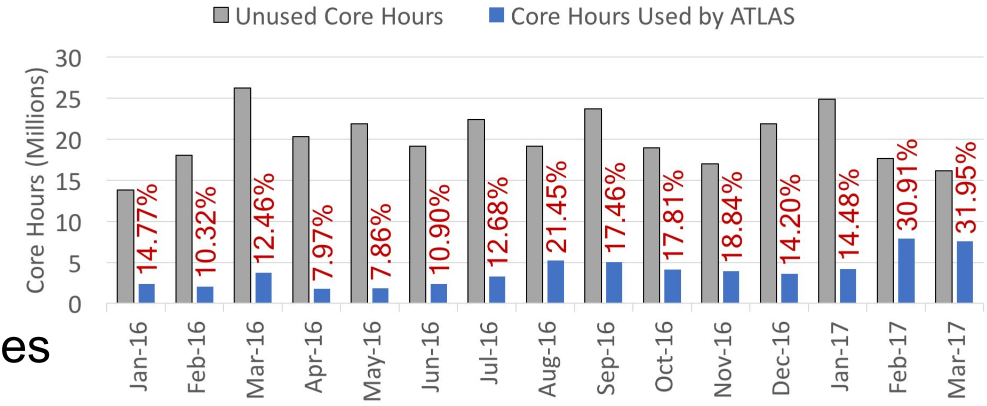
Understanding Backfill Slot Availability







- Efficiency: fraction of core-hours utilized by the PanDA Brokers to Titan's total backfill availability.
- Mean BF availability: 691 worker nodes for 126 minutes.
- Up to 15K nodes for 30-100 minutes
- large margin of optimization



Evolution of PanDA Deployment on Titan

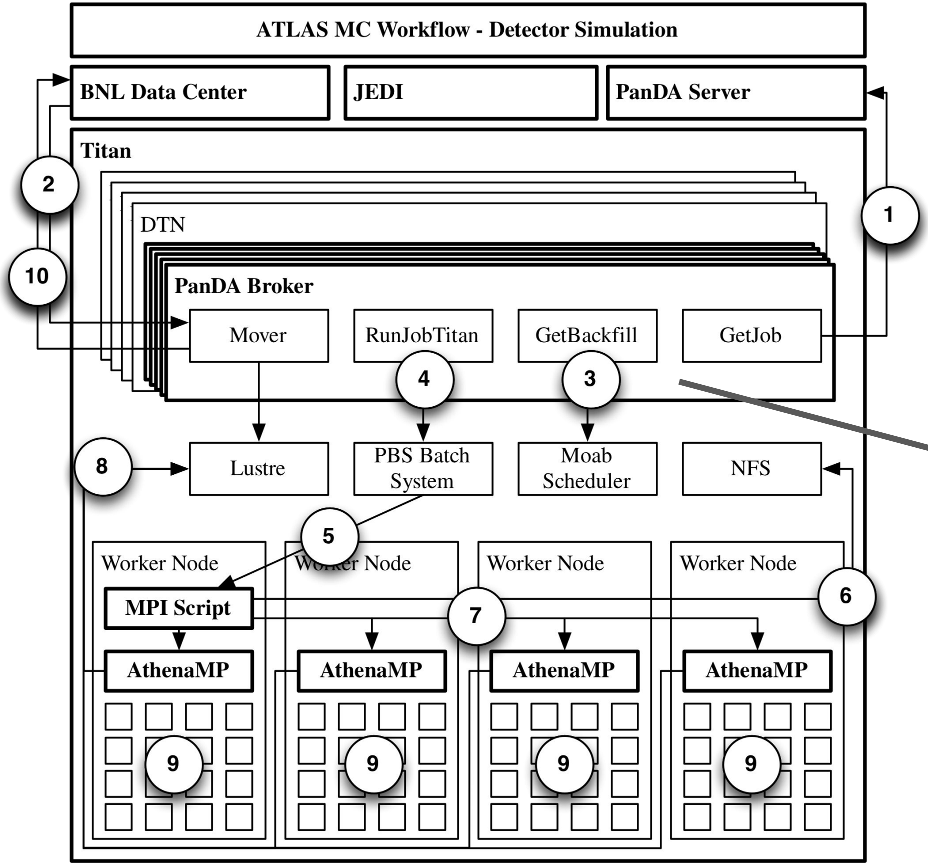
Static binding and sizing of tasks and jobs:

- Detector simulations of production ATLAS MC workflow
- 100 events per job (as opposed to 1000 for jobs on grid resources)
- 1 job; 1 WN; 16 AthenaMP; 100 events.

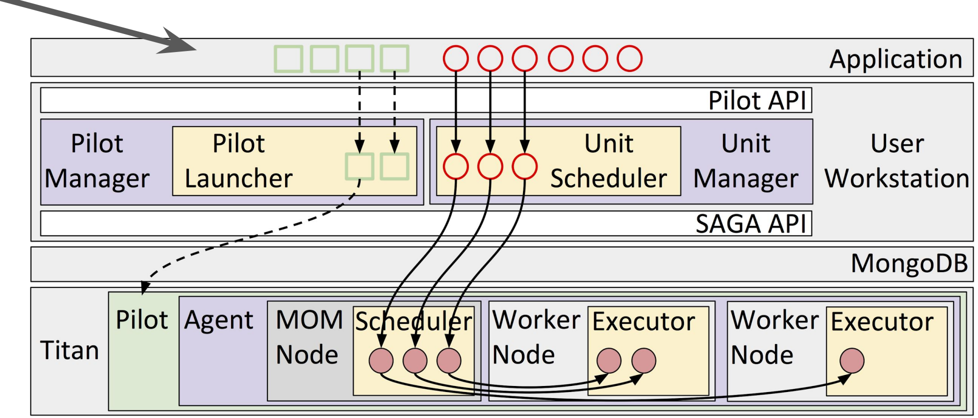
Benefits of implementing the pilot abstraction on Titan:

- Efficiency:
  - No assumption required about the number of events per simulation.
  - Jobs can be streamed to Titan when available.
  - Better walltime utilization via multiple execution generations on pilot(s).
- Workload heterogeneity:
  - No need of tailoring MPI scripts to the specifics of each type of workload.
- Resource heterogeneity:
  - No need of implementing tailored support for each HPC machine flavour.

PaNDA and Next Generation Executor (NGE)

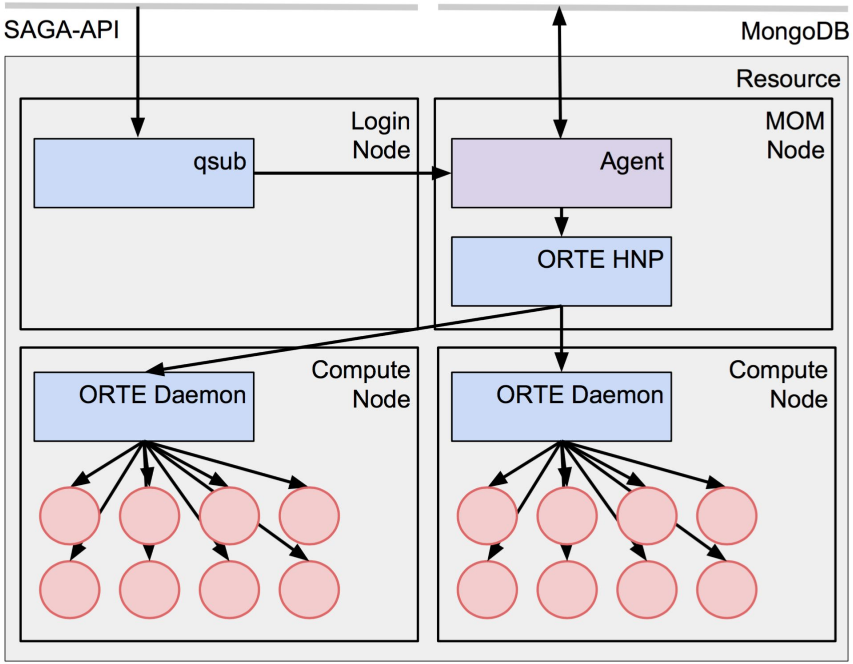


- NGE behaves like a resource:
  - ○ Exposes capabilities and availability acquired via pilots.
- Separation of concerns NGE independent from:
  - ○ PanDA server
  - ○ Type of workload



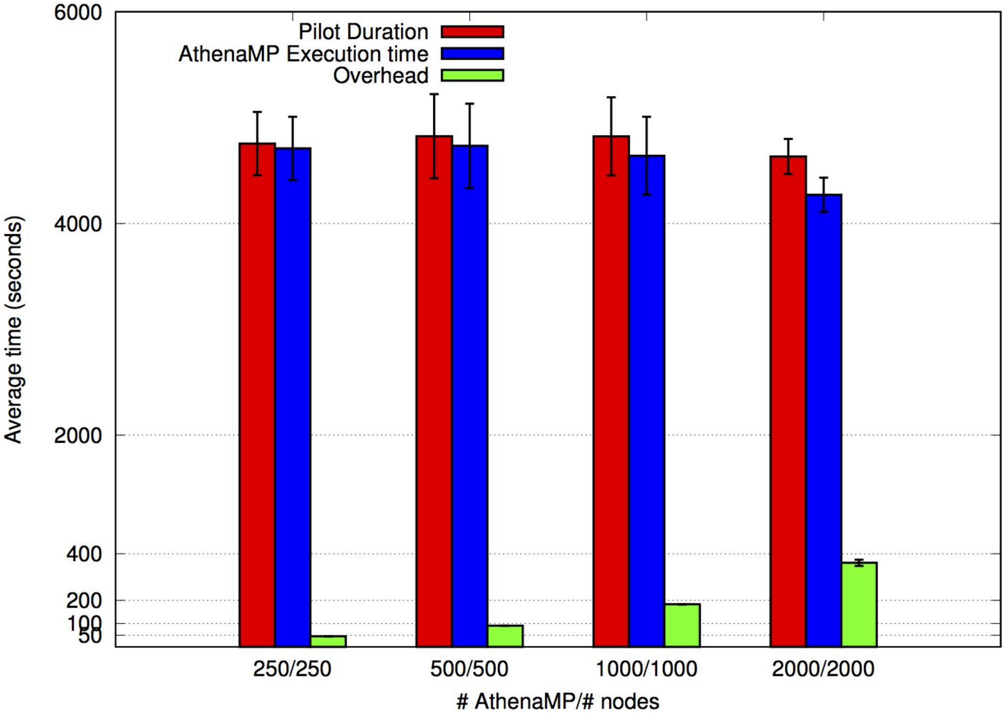
NGE Agent: ORTE-LIB

- ORTE: **O**pen **R**un**T**ime **E**nvironment
  - ○ Isolated layer used by Open MPI to coordinate task layout
  - ○ Runs a set of daemons over compute nodes
  - ○ No ALPS concurrency limits
- Supports multiple tasks per node
  - ○ Uses library calls instead of `orterun` processes
  - ○ No central fork/exec limits
  - ○ Shared network socket
  - ○ (Hardly) no central file system interactions



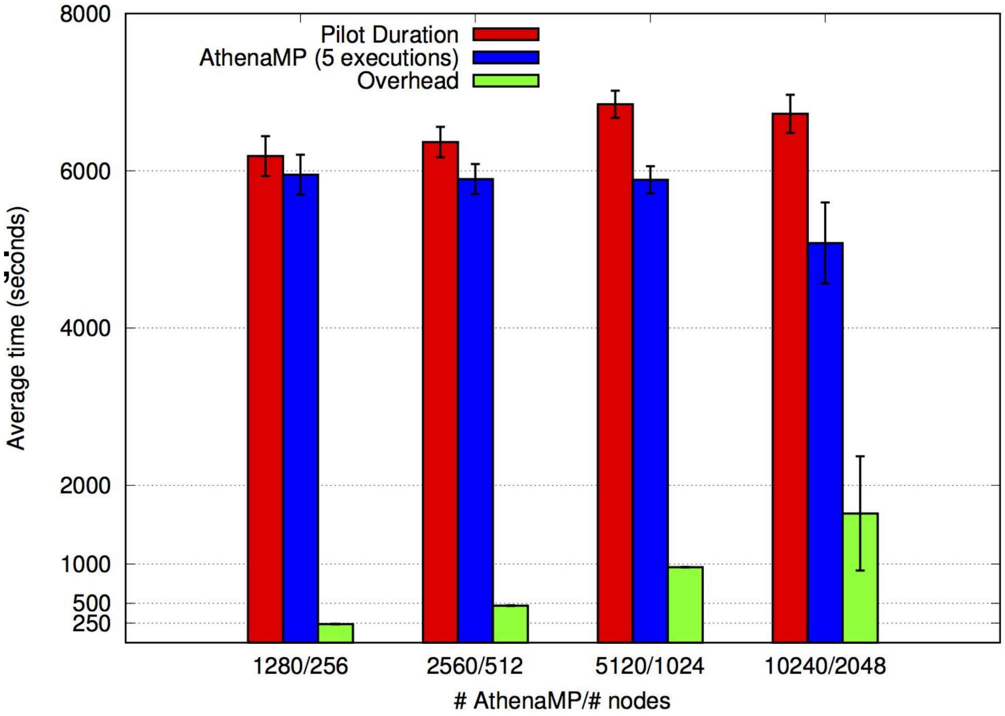
Experiments: Weak Scalability

- Each AthenaMP simulates 100 events in the ATLAS detector.
- One instance of AthenaMP per Titan's worker node; 16 Geant4 simulators per node.
- The overhead grows with the number of units but its growth is less than linear.



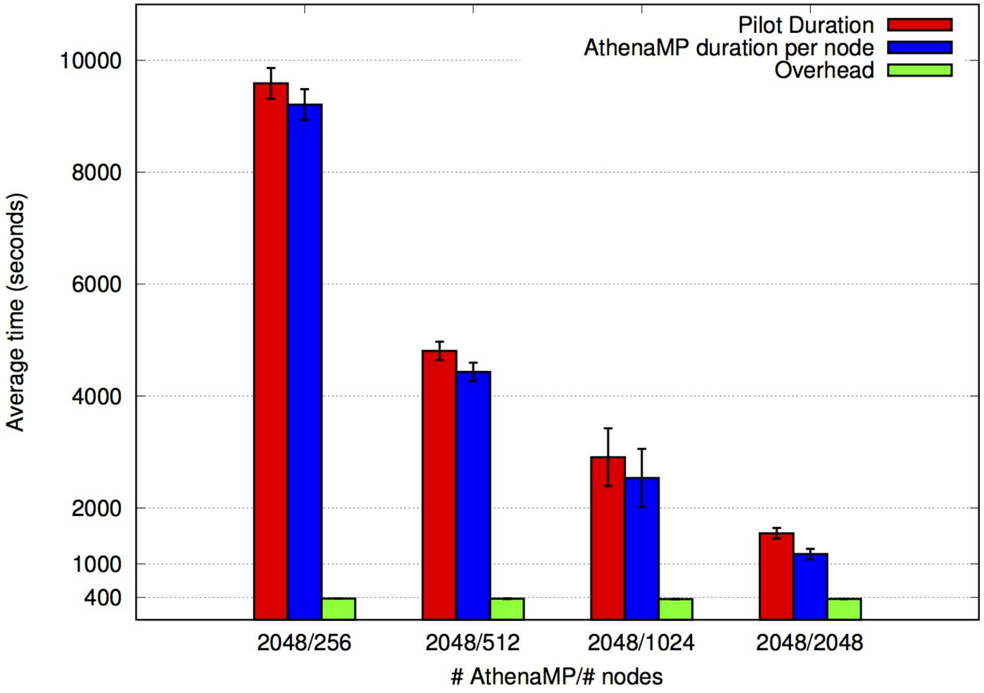
Experiments: Weak Scalability, Multiple Generation

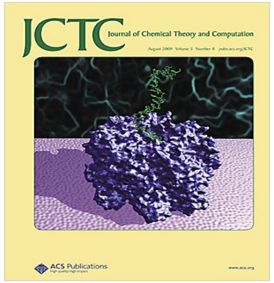
- Each AthenaMP simulates 100 events in the ATLAS detector.
- Five sequential instances of AthenaMP per Titan's worker node; 16 Geant4 simulators per node.
- Stress NGE components by creating concurrency between starting and ending units.
- The growth of the overhead is still less than linear.

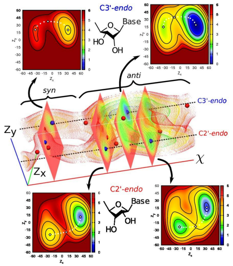


Experiments: Strong Scalability

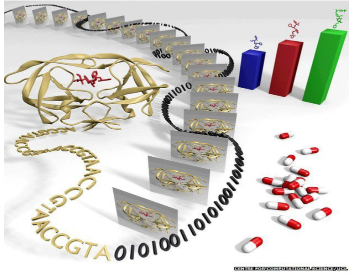
- Each AthenaMP simulates 100 events in the ATLAS detector.
- Each pilot is bounded with 2048 units.
- Smaller the pilot size, larger the number of sequential AthenaMP instances that are run by the pilot.
- The overhead remains constant while the pilot duration decreases.

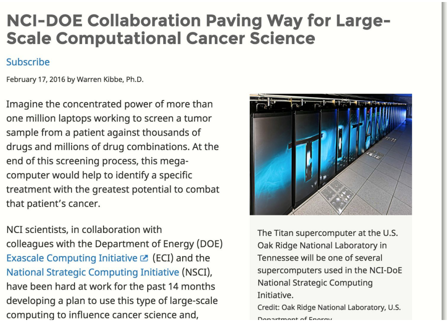



JCTC
Journal of Chemical Theory and Computation
August 2009 Volume 5 Number 8 pubs.acs.org/JCTC
ACS Publications
High quality. High impact.
www.acs.org



Biomolecular Simulations: Ensemble Execution





NCI-DOE Collaboration Paving Way for Large-Scale Computational Cancer Science

Subscribe

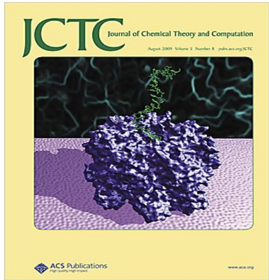
February 17, 2016 by Warren Kibbe, Ph.D.

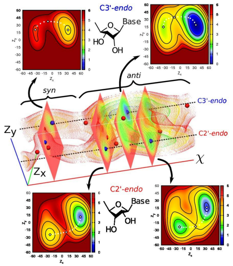
Imagine the concentrated power of more than one million laptops working to screen a tumor sample from a patient against thousands of drugs and millions of drug combinations. At the end of this screening process, this mega-computer would help to identify a specific treatment with the greatest potential to combat that patient's cancer.

NCI scientists, in collaboration with colleagues with the Department of Energy (DOE) Exascale Computing Initiative (ECI) and the National Strategic Computing Initiative (NSCI), have been hard at work for the past 14 months developing a plan to use this type of large-scale computing to influence cancer science and,

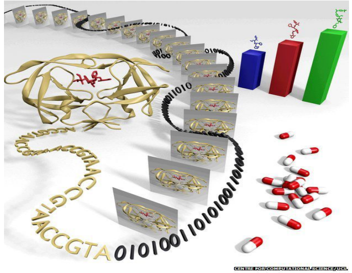
The Titan supercomputer at the U.S. Oak Ridge National Laboratory in Tennessee will be one of several supercomputers used in the NCI-DoE National Strategic Computing Initiative.

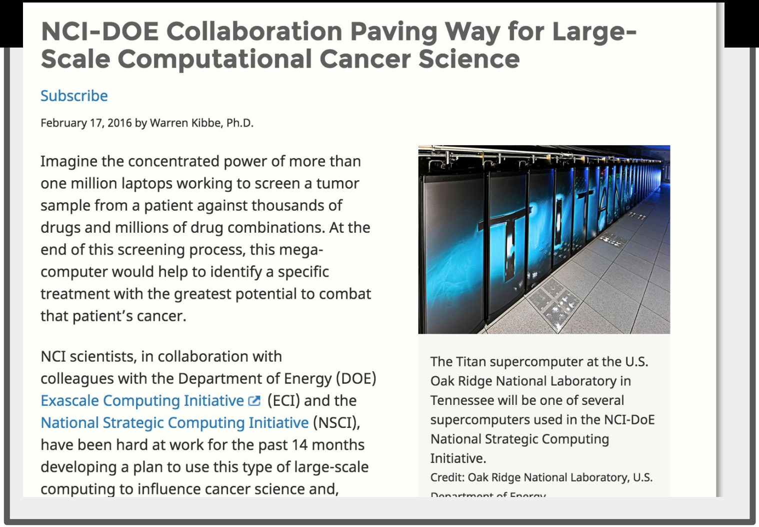
Credit: Oak Ridge National Laboratory, U.S. Department of Energy


JCTC Journal of Chemical Theory and Computation
August 2009 Volume 5 Number 8 pubs.acs.org/JCTC
ACS Publications



Biomolecular Simulations: Ensemble Execution





NCI-DOE Collaboration Paving Way for Large-Scale Computational Cancer Science

Subscribe

February 17, 2016 by Warren Kibbe, Ph.D.

Imagine the concentrated power of more than one million laptops working to screen a tumor sample from a patient against thousands of drugs and millions of drug combinations. At the end of this screening process, this mega-computer would help to identify a specific treatment with the greatest potential to combat that patient's cancer.

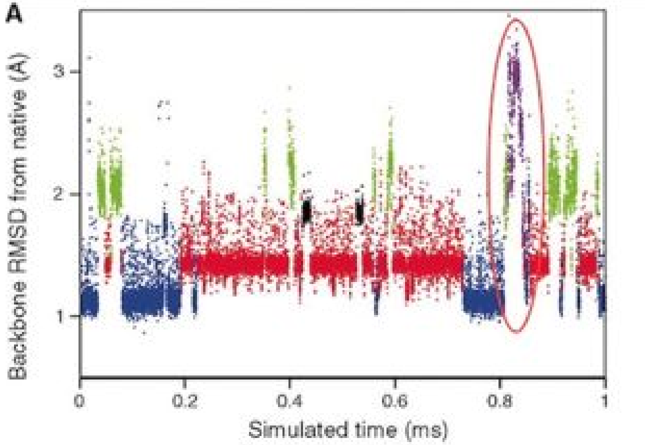
The Titan supercomputer at the U.S. Oak Ridge National Laboratory in Tennessee will be one of several supercomputers used in the NCI-DoE National Strategic Computing Initiative.

Credit: Oak Ridge National Laboratory, U.S. Department of Energy

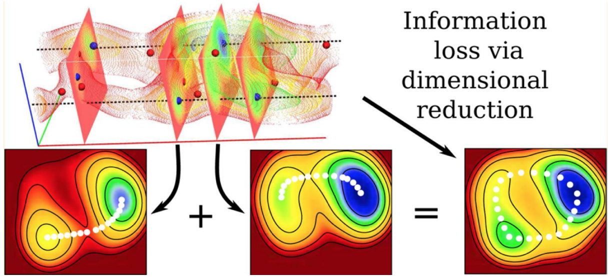
NCI scientists, in collaboration with colleagues with the Department of Energy (DOE) Exascale Computing Initiative (ECI) and the National Strategic Computing Initiative (NSCI), have been hard at work for the past 14 months developing a plan to use this type of large-scale computing to influence cancer science and,

New type of HTC: Bio-MD Simulations (BMS)

- **Convergence: Decompose a traditional HPC problem into an HTC problem!**
- Larger biological systems
  - Typically solved by weak scaling
- Long time scale problem
  - Typically solved by strong scaling
- **Gap between weak and strong scaling** capabilities will grow with "New" Moore's Law.
- Need advances in Algorithms to address this gap
  - Scaling challenges are more than single-partition strong and weak scaling

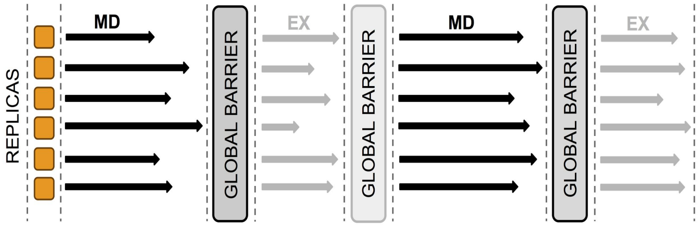


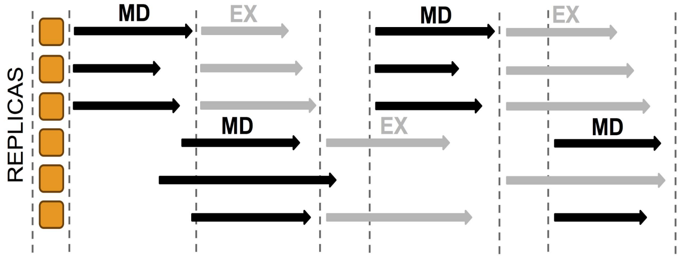
BPTI, 1ms MD ~3 months on Anton (Shaw *et al*, Science 2010)

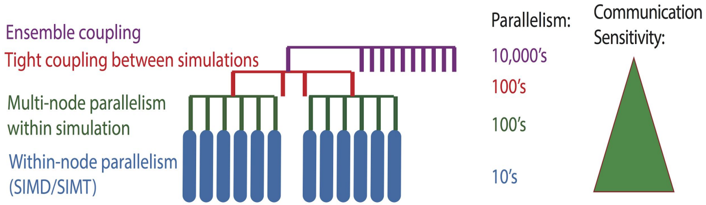


The Power of Many: Ensemble Methods

- Many **sampling problems** formulated as ensemble methods/algorithms
- Ensemble members often interact.
  - Not a “bag-of-tasks” abstraction.
  - Replica-exchange, Adaptive Markov State Models, Adaptive biasing...
- Different degrees and levels of coupling between ensembles members
- A HTC problem on HPC, but traditionally HPC optimized for single large job(s)!

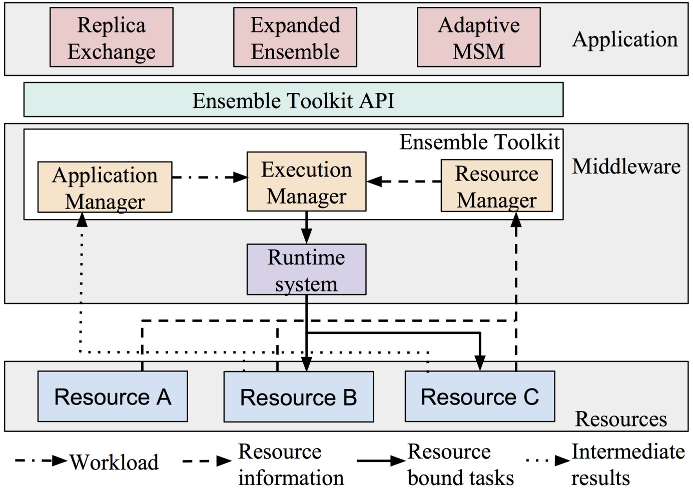


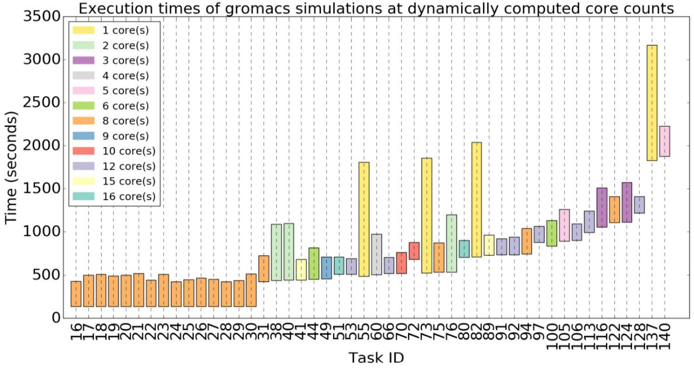



Ensemble coupling
Tight coupling between simulations
Multi-node parallelism within simulation
Within-node parallelism (SIMD/SIMT)
Parallelism:
10,000’s
100’s
100’s
10’s
Communication Sensitivity:

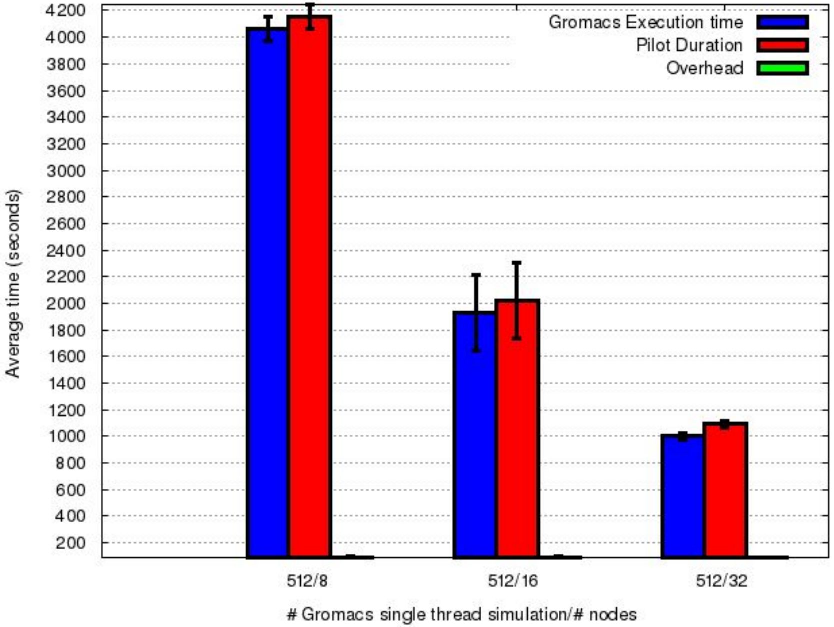
RADICAL-Cybertools: Ensemble Toolkit (EnTK)

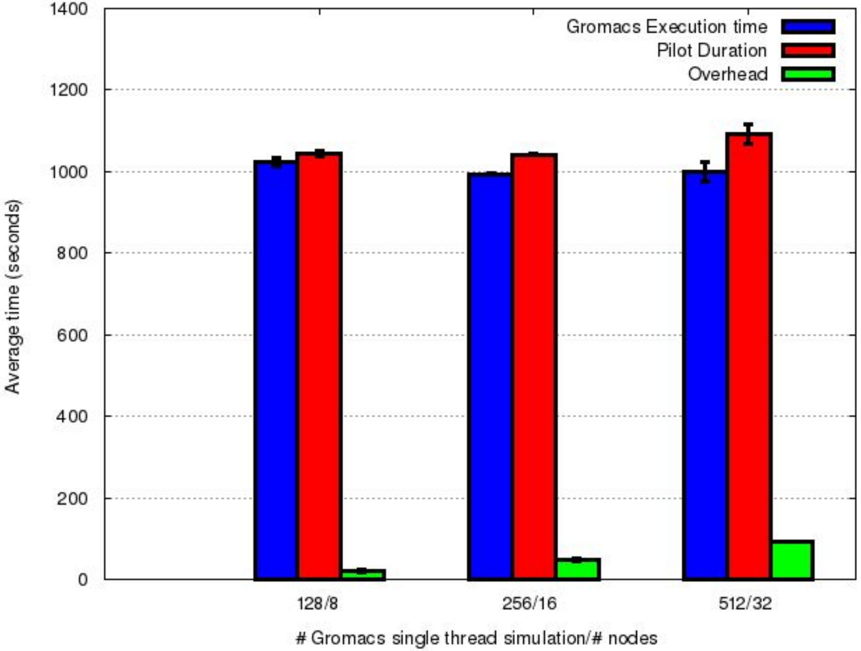
- Ensemble-Member = task = **Execution Unit**
  - Multi-node, sub-node, MPI/non-MPI...
  - Simulation, Analysis, ..
- **AIMES Execution Model**
  - Support for heterogeneous tasks
  - Adaptive Workload: Tasks and/or relations between tasks changes, or unknown *a priori.*
- Multiple dimensions of scalability:
  - Concurrency: O(10K) tasks
  - Task size: O(1)-O(1,000) cores
  - Launch: O(100+) tasks per second
  - Task duration: O(1)-O(10,000) seconds





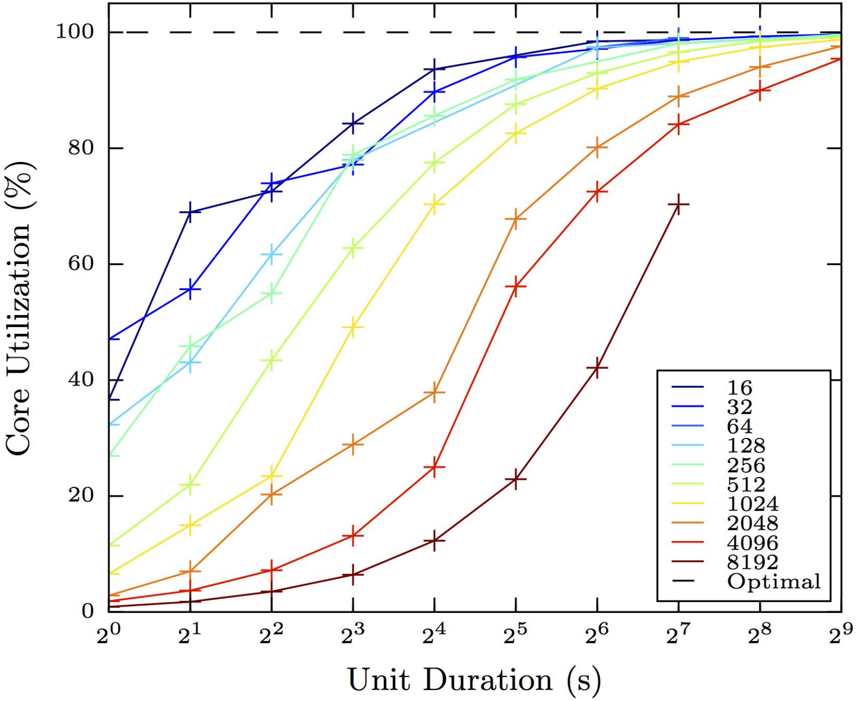
Many Gromacs on Titan: Close to linear scaling

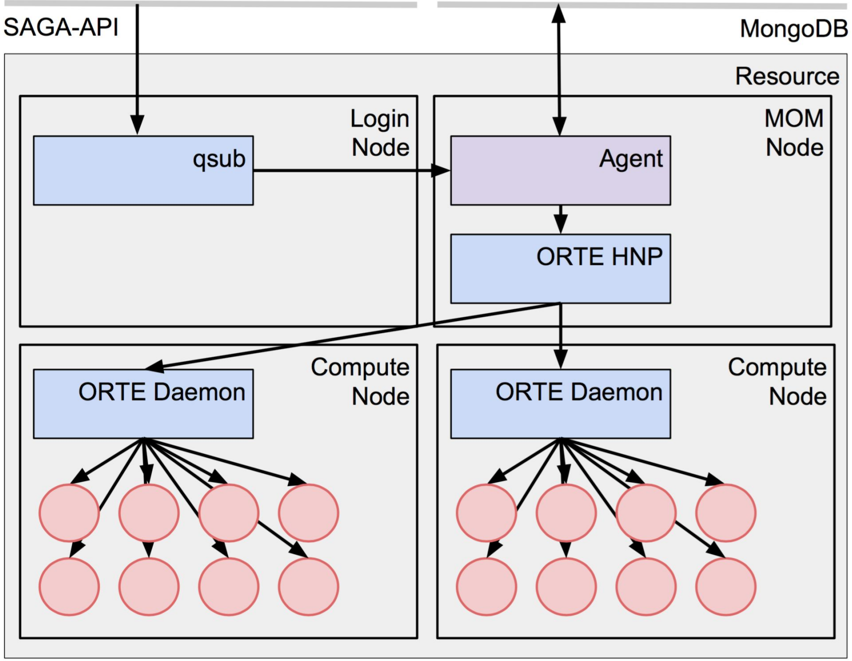




- Almost perfect scaling for O(1000)
- Ongoing work to support O(1000)-O(10,000) Gromacs currently.
- Need is for O(100,000). Challenges at multiple levels.

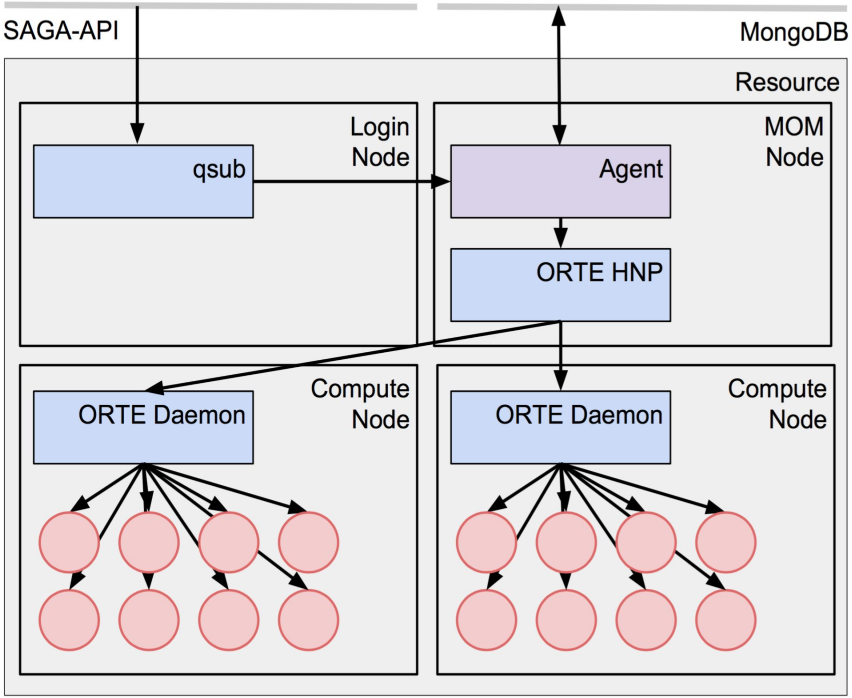
Experiments: ORTE-LIB





Challenges of O(100K+) Concurrent Tasks

- Information management!
- Agent communication layer (ZMQ) has limited throughput
  - bulk messages
  - separate message channels
  - code optimization
- Agent scheduler (node placement) scale well with number of cores
  - bulk operations
  - better scheduling algorithms
  - code optimization
- Collecting complete jobs is just as hard as spawning new ones



Summary:

- PanDA deployment of Titan shows the potential of D-HTC at unprecedented scale and for production workload.
- NGE enables the generalization and optimization of PanDA deployment on the based of a unified conceptual framework.
- Traditional HTC and HPC distinctions are a by-product of specific cyberinfrastructure implementations; discussions around software is misleading!
- Demonstrate how **abstractions**, **execution models** and multi-level support unify HTC, D-HTC, HT-HPC and HPC under the same conceptual framework enabling:
    - Design and development of general-purpose middleware.
    - Management of workload and resource heterogeneity and dynamism
    - Analytical understanding of workload, middleware, and resource behavior.